ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SONIDO E IMAGEN

Título del proyecto:

DESARROLLO DE UN SISTEMA DE VIDEOCONFERENCIA MEDIANTE WebRTC

Josu Ibarrola Lerga
Tutor: Marko Galarza Galarza
Pamplona, 9 de Septiembre de 2016

-ÍNDICE-

1. INTRODUCCIÓN

Vivimos en un mundo que avanza constantemente. A lo largo de la historia hemos ido progresando y adquiriendo conocimientos tecnológicos nuevos que nos han permitido cubrir necesidades y conseguir una calidad de vida mejor. Estos avances tecnológicos nos han situado, a día de hoy, en la era de la informática y la información.

La tecnología informática ha vivido una auténtica revolución que ha afectado tanto a nuestras costumbres como a nuestro modo de vida. Y es que internet está presente casi en todas nuestras facetas del día a día. Desde leer los correos del trabajo, a reservar unas entradas para disfrutar de cualquier espectáculo en nuestro tiempo libre.

Las páginas web también han ido evolucionando conforme evolucionaban las tecnologías de su desarrollo, HTML, CSS, JavaScript… siendo cada vez más funcionales y completas.

El auge de las redes sociales, blogs, WhatsApp… como forma de expresarse ha hecho que mucha gente quisiera crear su propio blog o página web. Esto propició que se desarrollaran las formas para realizar complejas páginas web de la manera más sencilla posible. Todo esto ha sido posible gracias al software de código libre, cuyo código fuente está accesible a todo el público. Cualquiera puede crear sus aplicaciones cogiendo como base el código libre desarrollado. Pudiendo modificar dicho código y sin la necesidad de obtener ningún tipo de licencia o permiso que suponga un desembolso económico.

En este proyecto desarrollaremos un sistema de videoconferencia basado en WebRTC, que es un ejemplo más del desarrollo de software de código abierto que nos permite realizar acciones como videollamadas de una manera sencilla y eficaz sin tener que pagar ningún tipo de licencia de uso.

2. OBJETIVO

El objetivo del proyecto a realizar consiste en el desarrollo de un sistema de videoconferencia mediante WebRTC. Para realizar dicho desarrollo crearemos una página mediante el lenguaje de programación HTML5 y la tecnología que nos brinda WebRTC, todo ello dentro del marco de trabajo de Bootstrap, consiguiendo así una página web apta para cualquier dispositivo de visualización que se utilice.

Debido a que se trata de una tecnología nueva que el autor desconoce, para poder realizar correctamente el sistema de videoconferencia primero realizaremos un estudio de WebRTC, ¿Qué es? ¿Cómo funciona? Para posteriormente poder plasmar los conocimientos adquiridos en el ejemplo de videoconferencia que estará dentro de nuestra página web.

Para realizar nuestro desarrollo del sistema de videoconferencia necesitaremos emplear varias tecnologías. Cada una cumplirá su función y la combinación de todas ellas nos permitirá la creación del nuestra página web, desde dónde realizaremos las videoconferencias. Estas tecnologías son: HTML, CSS, Bootstrap, JavaScript, WebRTC, jQuery y Node.js

Finalizaremos contando el futuro próximo de WebRTC, así como exponiendo las conclusiones extraídas en la realización de este proyecto.

3. TECNOLOGÍAS

3.1 WebRTC

Figura 1. Logotipo de WebRTC.

WebRTC significa web de comunicación en tiempo real. Se trata de una API que ha sido y sigue siendo desarrollada por W3C (World Wide Web Consortiun).

El propósito principal de los creadores de WebRTC es la creación de una plataforma común en la cual poder enlazar diferentes dispositivos tales como: teléfono, televisión, ordenador… mediante el uso compartido de datos *Peer-To-Peer*.

Antes de la aparición de WebRTC dicha comunicación suponía una barrera difícil de flanquear debido a las costosas tecnologías, entre ellas servidores específicos que eran necesarios, así como la obtención de diferentes licencias que utilizan los distintos dispositivos que queremos comunicar entre sí de una forma directa. Actualmente la tecnología RTC (Real Time Comunication) ya está siendo usada por muchos servicios web tales como: Facebook, que utiliza Skype para sus vídeos; Hangouts de Google o el propio Skype. Pero estos servicios requieren de aplicaciones nativas, *plugins* o descargas.

Para hacer frente a estos problemas surgió WebRTC. Su principal característica es el enlace entre dispositivos de forma directa mediante el uso compartido de datos *Peer-To-Peer* sin ningún tipo de *plugin*, directamente entre navegadores. El hecho de no necesitar *plugin* alguno es una gran ventaja por los inconvenientes que estos pueden acarrear, como por ejemplo el hecho de tener que descargarlos, instalarlos y posteriormente actualizarlos cada vez que una nueva versión haya salido. Esta tarea puede resultar compleja para el usuario con conocimientos informáticos medios, además de acarrear otros problemas como pueden ser: ser más propenso a errores, difíciles de implementar, de depurar, más el coste económico de adquirir su licencia para su posterior uso.

Compatibilidad de navegadores.

No todos los navegadores son compatibles con WebRTC. Google Chrome, Mozilla Firefox con sus versiones tanto de escritorio como sus versiones de Android para el móvil son los navegadores que soportan WebRTC. Este punto es muy importante ya que la función de WebRTC es establecer la conexión directamente entre navegadores. Al ser una tecnología reciente, las versiones menos recientes de los navegadores no soportan WebRTC y tampoco lo hacen todos los navegadores. Las nuevas versiones de Chrome y Firefox son los navegadores que mejor rendimiento tienen con esta tecnología, mientras que otros como Internet Explorer de Microsoft y Safari de Apple solo funcionan instalando un *plugin* específico. Estos navegadores por sí solos no tienen soporte para WebRTC.

Por esta razón, este proyecto se realizará utilizando el navegador Chrome. Con el tiempo se espera la estandarización del uso de WebRTC para los distintos navegadores, pero de momento es necesaria la utilización de estos navegadores para el correcto funcionamiento de WebRTC.

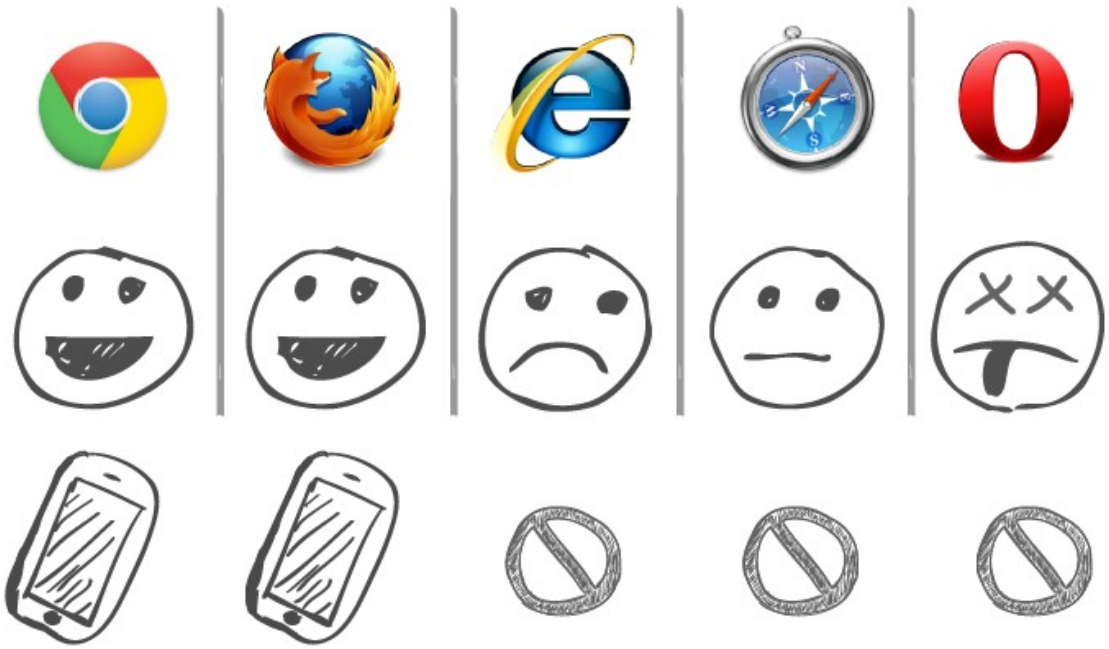


Figura 2. Compatibilidad de los navegadores.

La señalización:

Pese a que WebRTC se encarga de mantener el flujo de datos entre navegadores de forma directa, *Peer-To-Peer*, son necesarios mecanismos para coordinar la comunicación y enviar mensajes de error en el caso de que fuera necesario. Estas funcionalidades no están incluidas en las API (Application Programming Interface) de WebRTC, por lo que es necesario el proceso de señalización. Sus principales características son el control de la sesión, de la red y los medios de información. Por lo tanto, los desarrolladores pueden usar el protocolo de mensajería que estimen oportuno, algunos ejemplos pueden ser SIP (Session Initiation Protocol) o XMPP (Extensible Messagin and Presence Protocol). Que cumplirán las tareas de:

- Mensajes de control.
- Configuración de la red con el mundo exterior. Obtener la IP (Internet Protocol) y el Puerto del dispositivo para su posterior uso.
- Funciones de los medios de comunicación, como la regulación de codecs y resoluciones de ambos navegadores.

Por eso es tan importante, ya que sin señalización no podríamos obtener los datos que nos son imprescindibles para poder establecer la comunicación *Peer-To-Peer*, que es la base de WebRTC.

Funcionamiento:

WebRTC implementa tres API:

- *Mediastream* (se conoce también como *GetUserMedia()*): cuya principal característica es la obtención de acceso al flujo de datos como podrían ser la cámara del usuario y su micrófono.

- *RTCPeerConection*: Establecer las llamadas de vídeo y audio con instalaciones para el cifrado y la gestión del ancho de banda.

- *RTCDatachanel*: Se encarga de la comunicación *Peer-To-Peer* de datos genéricos.

Media Stream (GetUserMedia()):

Se encargar de los flujos de medios de comunicación sincronizados. Como en nuestro caso de la entrada de vídeo de la cámara y la entrada de audio mediante el micrófono. Cada *MediaStream* cuenta con una entrada, que normalmente será un MediaStream generada por la instrucción *navigator.GetUserMedia()*, en nuestro caso los datos de audio y vídeo. Y una salida que será el *stream.multimedia* sincronizado que se pasa a la siguiente API que es *RTCPeerConnection.*

Esta API tiene en cuenta tres parámetros:

- Las limitaciones de un objeto, que son marcadas por los diferentes navegadores web. De ahí la importancia de escoger el navegador adecuado para utilizar WebRTC.

- La devolución a su llamada con éxito, en cuyo caso nos devuelve la *MediaStream* sincronizada.

- La devolución a su llamada sin éxito, en cuyo caso nos devuelve un objeto de error.

Las restricciones que se pueda encontrar esta API se implementan desde los navegadores (Firefox, Chrome y Opera), y suelen ser para restringir los valores de las resoluciones de vídeo, la relación de aspecto, modo de orientación (cámara frontal o posterior), su velocidad, altura y anchura para *GetUserMedia()* y *RTCPeerConnection*.

***RTCPeerConnection*:**

Es el auténtico motor de WebRTC, se encarga de la comunicación RTC y de que la transmisión de datos entre pares se realice de forma estable y eficiente, incluso en redes que no son del todo fiables. Para eso utiliza una serie de codecs y protocolos que realizan una labor enorme para que las comunicaciones en tiempo real sean posibles. De

esta manera protege a los desarrolladores de las dificultades y complejidades que se presentan para realizar dicha unión. Entre sus labores se encuentran:

- Ocultación de pérdida de paquetes.
- Cancelación del Eco.
- La adaptabilidad del ancho de banda.
- Buffer de fluctuación mecánica.
- Control de ganancia automática.
- Reducción y supresión de ruido
- Imagen de Limpieza.

Para las distintas aplicaciones, WebRTC necesita el uso de servidores con los cuales solventará problemas como:

- Necesidad de intercambiar datos como nombres.
- Información de la red de intercambio.
- Datos sobre los medios de comunicación como el formato de vídeo y la resolución.
- Poder atravesar las puertas de enlace NAT (Network Address Translation) así como los firewalls.

Estos servidores son necesarios para WebRTC cumpla su cometido de:

- Detección del usuario y de la comunicación.
- Las funciones de la señalización
- Pasar por las diferentes NAT y Firewalls transversales.

- Restaurar la retrasmisión en el caso de que la comunicación *Peer-To-Peer* fallara.

Para realizar dicha unión contamos con los servidores ICE (Interactive Connectivity Establishment), STUN (Session Traversal Utilities for NAT) y TURN (Traversal Using Relays around NAT). El protocolo que siguen es el siguiente:

ICE es un marco de trabajo para unir a dos usuarios, como por ejemplo podrían ser dos clientes del chat de vídeo. Su objetivo es comunicar dos terminales entre sí y para ello probará distintas rutas. Lo primero que intentara será conectar directamente a los dos compañeros a través de UDP (User Datagram Protocol)con la menor lactancia posible. Para ello recurre a los servidores STUN, cuya única tarea en este caso será la de habilitar a un cliente que se encuentra detrás de una NAT mediante la obtención de su dirección física y del puerto.

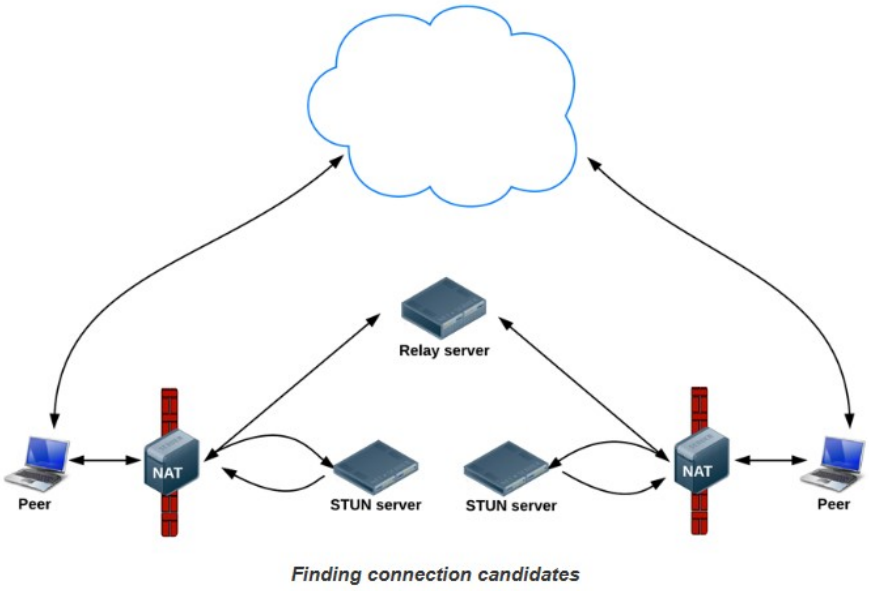


Figura 3. Conexión mediante servidores STUN.

Si fallase el modo de conexión mediante UDP, ICE lo intentaría de nuevo mediante TCP (Transmission Control Protocol), a través de http y https. Si la conexión directa fallara a causa de la NAT o de los diferentes firewalls, ICE recurriría a el servidor de retrasmisión intermediario TURN, que se encargaría de coordinar la información así como de atravesar cortafuegos y NAT simétricos.

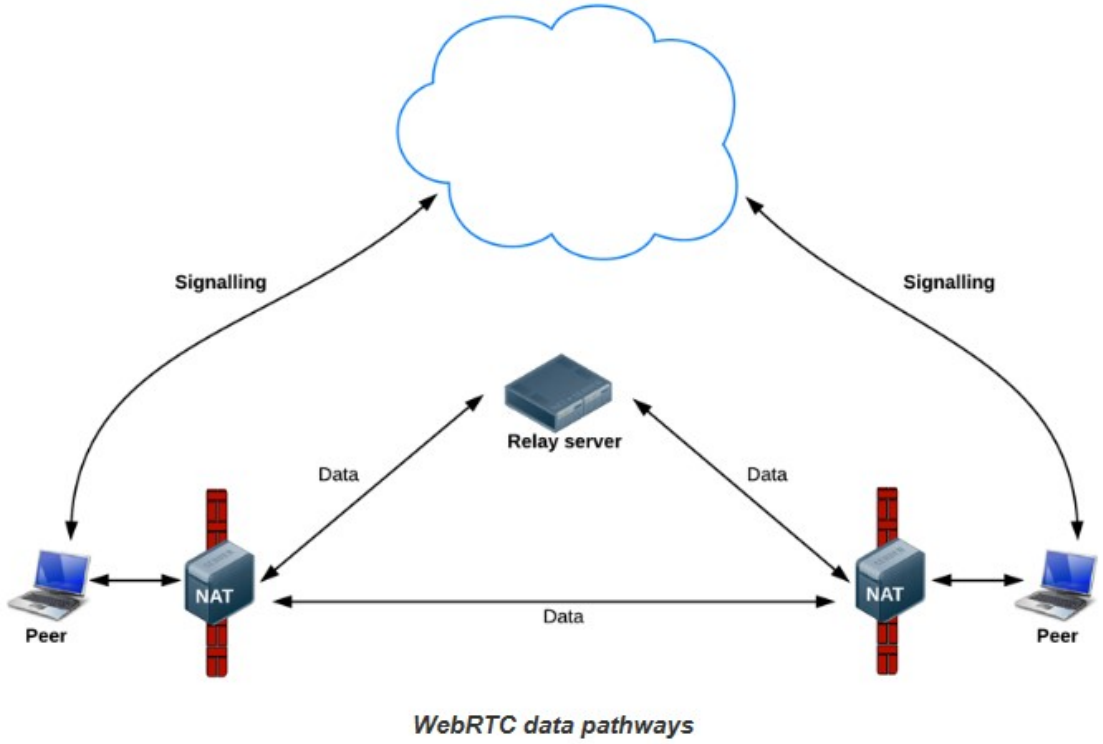


Figura 4. Conexión mediante servidor TURN.

RTCDatachanel:

WebRTC no solo es capaz de soportar la comunicación en tiempo real de vídeo o audio, también soporta la comunicación en tiempo real para otro tipo de datos. Esta API nos permite el intercambio de datos arbitrarios mediante comunicación *Peer-To-Peer* con un alto rendimiento y con baja latencia.

Nos sirve muy bien para juegos multijugador, chat de texto, aplicaciones remotas, transferencia de archivos… puede proporcionarnos una comunicación flexible y potente *Peer-To-Peer* debido a que cuenta varias características para aprovechar las ventajas que nos proporción *RTCPeerConection*. Por ejemplo: canales múltiples simultáneos con establecimiento de prioridades, aprovecha la sesión ya iniciada de *RTCPeerConection*, capacidad de seguridad (DTSL), el control de la congestión…

La comunicación se realiza directamente entre los navegadores, por lo que es más rápida. En el hipotético caso de que tendríamos que utilizar el servidor TURN debido a los problemas de conexión, seguiría siendo rápida.

Seguridad en WebRTC:

Siempre que se establece una comunicación existe el riesgo de que la seguridad de dicha transmisión se pueda ver comprometida por determinadas aplicaciones de comunicación en tiempo real o *plugins*. Los principales problemas a los que se enfrenta la seguridad de WebRTC son los siguientes:

- Una aplicación puede interceptar datos o medios de comunicación sin cifrar entre las diferentes rutas que los navegadores tienen entre sí o entre las rutas de los navegadores y los servidores.

- Algunas aplicaciones pueden tener la capacidad primero de grabar y posteriormente distribuir un archivo tanto de vídeo como de audio sin el conocimiento del usuario.

- Al instalarnos un *plugin* u otra aplicación aparentemente inocua, se podrían instalar distintos virus y malware.

Dada la complejidad de WebRTC y la cantidad de datos, archivos, información… que es capaz de transmitir en tiempo real, han sido necesarias tomar varias medidas de protección entre las que destacamos:

- El uso de protocolos seguros como son DTSL (Datagram Transport Layer Security) y SRTP (Secure Real-time Transport Protocol)

- Todos los componentes de WebRTC, incluidos los mecanismos de señalización están cifrados.

- Los componentes de WerRTC no son *plugin*, por lo que no precisan de instalación. Eliminando de ese modo el riesgo que eso supone para nuestra comunicación. Dichos componentes se ejecutan en el recinto de seguridad del propio navegador y son actualizados cada vez que el navegador es actualizado.

- Para poder acceder tanto a la cámara como al micrófono se debe conceder la autorización desde el cliente de manera explícita, y una vez que ya se están ejecutando, se puede ver claramente a través de la interfaz del usuario.
  [1, 2, 3, 4, 11]

3.2 BOOTSTRAP

Figura 5. Logotipo de Bootstrap.

Se trata de un framework que fue creado y desarrollado por los ingenieros de Twitter, que posteriormente liberalizaron su código, siendo desde entonces de código abierto.

Este framework nos permite crear interfaces de web utilizando HTML, JavaScript y CSS. Su principal característica es que utiliza una técnica de desarrollo y diseño que es conocida como "*responsive design*". Esto quiere decir que tiene un diseño adaptativo. Gracias a esta característica el interfaz web que construyamos bajo el marco de desarrollo de Bootstrap se podrá visualizar de forma óptima a las diferentes resoluciones en las queramos visualizar nuestra interfaz. Bootstrap nos permite diseñar teniendo en cuenta desde qué dispositivos vamos a visualizarlo, por lo que responderá automáticamente adaptándose. Se puede hacer el ajuste automáticamente como hacen otros frameworks y que el contenido de nuestra interface de adapte al tamaño de la pantalla, reduciéndose o agrandándose según se reduzca o amplíe el tamaño del dispositivo donde lo vamos a visualizar. Esto suele traer problemas, sobre todo en los dispositivos de visualización más pequeños donde los textos y las fotos tienden a amontonarse, reduciendo así su visibilidad.

Figura 6. Icono de responsividad de Bootstrap.

Con Bootstrap tenemos la capacidad de programar y diseñar nuestro interfaz cambiando el orden en el que se muestran nuestros contenidos, o directamente tenemos la capacidad de suprimir ciertos elementos para que en determinadas resoluciones no se nos muestre y podamos ver el resto de nuestros contenidos de la forma más óptima. Esto lo consigue utilizando un simple sistema de rejillas. Este sistema divide toda la pantalla en 12 rejillas, por lo que tenemos referencias espaciales para la posterior organización del contenido. Por ejemplo, podemos poner que determinada sección nos rellene cuatro, ocupando un tercio de la pantalla, sólo en los dispositivos más grandes. Mientras que esa misma sección en los dispositivos más pequeños puede ocupar las doce, el ancho total de la pantalla. De ese modo podremos ver el contenido de nuestra página de la manera más óptima en los dispositivos de cualquier tipo de tamaño.

Figura 7. Sistema de rejillas de Bootstrap.

Todo depende del tamaño del dispositivo donde lo visualicemos. Lógicamente los dispositivos de visualización que más problemas generan son los smartphones debido a su pequeña pantalla. Por eso Bootstrap se define como un framework "*mobile first*", esto es, está pensado para programar desde pantallas más pequeñas a más grandes, asegurándonos de esta manera su correcta visualización en cualquier tipo de dispositivo independientemente del tamaño que este tenga.

Además Bootstrap cuenta con diferentes galerías propias, por lo que aparte de los elementos HTML comunes cuenta con decenas de elementos HTML personalizados como pueden ser el caso de botones, menús de navegación desplegables, formularios… también cuenta con componentes CSS o JavaScript avanzados.

Otra de sus características y ventajas, es que está construido para ser usado a todos los niveles, tanto en proyectos de poca envergadura diseñados por usuarios de poco nivel, como en proyectos de mayor envergadura y complejidad desarrollados por profesionales. Su manejo e implantación son sencillos y los diseños creados por Bootstrap suelen ser limpios, sencillos e intuitivos. [1, 2, 4, 10]

3.3 CSS.

Figura 8. Logotipo de CCS.

Se conoce como hoja de estilo en cascada. Es un lenguaje que se usa en combinación con HTML o XML. Las páginas de HTML necesitan una apariencia óptima para poder ser visualizadas correctamente y para que consten de una presentación más adecuada y más atractiva. Esto se consigue mediante CSS, que se ocupa de definir la presentación de un documento HTML. Por lo que de esta manera conseguimos separar el contenido de la forma. Pudiendo alterar de manera más sencilla la forma en la que queremos que se muestre posteriormente nuestro contenido. La siguiente imagen muestra una página web. En el lado de la izquierda se ha utilizado CSS y en el de la derecha no. Se puede observar claramente el resultado estético final dista mucho uno de otro pese a que el contenido de la página es el mismo.

Figura 9. La misma página utilizando CSS y sin usarlo.

CSS se encarga de obtener un control centralizado, por lo que podremos actualizar más fácilmente la presentación de un sitio web completo. Dado que un mismo archivo de CSS puede utilizarse para varios documentos, conseguimos una optimización del ancho de banda así como una reducción del código considerable.

Actualmente se encuentra en su tercera versión y se le conoce como CSS3.
[1, 2, 4, 9]

3.4 HTML

Figura 10. Logotipo de HTML.

HTML son las siglas de HiperText Markup Lenguage, lo que se traduciría como lenguaje marcado de hipertexto.

Se trata de un lenguaje de programación para la elaboración de páginas web. Este lenguaje nos permite etiquetar y ordenar diversos documentos dentro de una lista de una manera fácil y sencilla. Por lo que este lenguaje se basa en el uso de etiquetas que posteriormente son interpretadas por el navegador, mostrando posteriormente el resultado por la pantalla. Lo que observamos en el navegador es la interpretación que este ha realizado del documento HTML.

Se podría decir que HTML seria el esqueleto de nuestra página web, ya que dentro del archivo HTML se recurre con frecuencia a librerías de JavaScript como jQuery o a funciones de JavaScript que nos permitirán realizar operaciones más complejas en nuestra página.

Actualmente se encuentra en su quinta versión y se lo conoce como HTML5. Las mejoras introducidas son para mejorar el contenido sin tener que recurrir a *plugins*. [1, 2, 4, 8]

3.5 jQuery

Figura 11. Logotipo de jQuery.

Es un framework de JavaScript de código abierto que funciona como una biblioteca donde se almacenan un conjunto de habilidades que podemos utilizar sin tener que programarlas por qué ya fueron programadas y posteriormente probadas. Por eso el lema de jQuery es "*write less, do more.*" Que podría traducirse como "*escribir menos, hacer más*".

Gracias a esta colección de recursos podemos incorporar a nuestra página funcionalidades y efectos de distintos tipos que suelen servir para interactuar con nuestros elementos de la página, como validación de formularios, galerías de fotos dinámicas... Esta es una de sus características, que nos permite seleccionar elementos HTML de nuestra página de manera fácil y sencilla.

El hecho de que el código de estas aplicaciones ya esté escrito nos supone un enorme ahorro a la hora de programar nuestra propia aplicación. Enlazando jQuery a nuestra aplicación, podremos realizar la función que queramos hacer, reduciendo de ese modo de manera considerable el código de nuestra aplicación. A la par que conseguimos funcionalidades complejas de una forma muy sencilla. De esta manera nos ahorramos la extrema dificultad que supone programar ciertas funcionalidades de jQuery que ya se encuentran programadas para el uso de los programadores.

Existen otras librerías de JavaScript como pueden ser Angular.js o Zepto.js, pero hemos optado por utilizar jQuery ya que tiene más recorrido y su uso está muy extendido. Además de contar con muchos recursos y añadidos, es totalmente compatible con la mayoría de exploradores. [1, 2, 4, 6]

3.6 Node.JS

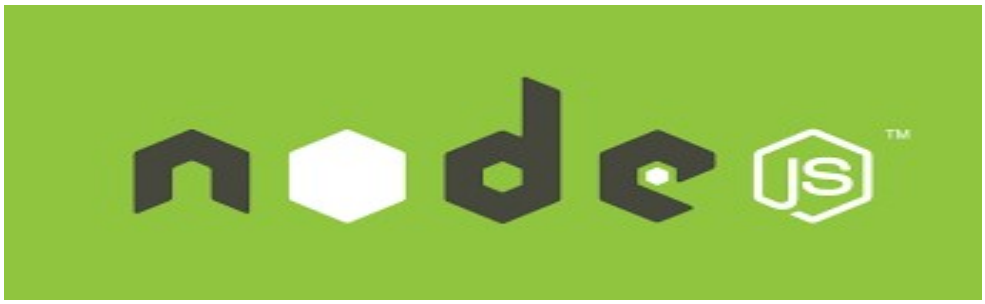


Figura 12. Logotipo de Node.js.

Node.js es una plataforma construida sobre el motor de JavaScript de Chrome llamado V8. Puede funcionar como un intérprete de los archivos de JavaScript. Para poder comprobar si nuestra página web funciona correctamente necesitaríamos el uso de un servidor, pero al realizar este proyecto de manera local es necesario instalar algún programa que realizara esa función para poder chequear el correcto funcionamiento de nuestro archivo de JavaScript que se encarga de la creación de la sala de videoconferencias.

Nuestra aplicación utiliza comunicación en tiempo real, Node.js es la mejor opción con la que contamos ya que a diferencia de otros servidores, donde mandamos un petición desde nuestra página y al recibir la respuesta nuestra página era recargada, Node.js está pensado para aplicaciones en tiempo real y no es necesario recargar la página al recibir la respuesta, el servidor responde automáticamente cuando llegue información nueva.

Aunque Node.js puede actuar como un servidor, al tratarse de una plataforma JavaScript extensible, está capacitado para realizar diferente acciones como pueden ser soportar el envío de ciento de miles de conexiones concurrentes. Un ejemplo de esto podrían ser las estadísticas de los videojuegos donde continuamente se están rastreando los datos de todos los jugadores para mostrar las estadísticas, necesitaríamos varios servidores para manejar tal cantidad de información, mientras que necesitaríamos un único servidor que ejecute Node.js para lograr el mismo fin. [1, 2, 4, 7]

3.7 JavaScript

Figura 13. Logotipo de JavaScript.

JavaScript es un lenguaje de programación que se usa para la creación de programas. Estos programas son capaces de realizar acciones específicas. Posteriormente pueden ser insertados en otros programas o en páginas web para que realicen su función.

Su principal característica es que se trata de un lenguaje de programación basado en acciones, entre las que destacan entre otras muchas la utilización de teclas, funciones que responden al movimiento del ratón, la descripción de objetos aperturas, cargas de páginas…

Como ocurre con jQuery, podemos tener un programa de JavaScript ya implementado y hacer una llamada desde nuestro propio programa consiguiendo de esta manera que nuestro código sea más simple y más corto. [1, 2, 4, 5]

4. DESARROLLO

El sistema de videoconferencias se encuentra dentro de la página web que hemos creado, por lo que empezaremos describiendo la página web.

4.1 ESTRUCTURA HTML

La página web constará de tres secciones. La página de inicio, otra página que contendrá información relacionada con WebRTC, donde se explicará de forma breve:

- La historia de la creación de WebRTC.
- La seguridad que emplea para hacer que sus envíos de información sean seguros.
- Las tecnologías y funciones de sus servidores.
- La compatibilidad de WebRTC con los diferentes navegadores donde poder emplearla.

La tercera página será donde creemos la sala de videoconferencias. Por lo tanto nuestra página web contará con tres archivos de HTML, dos hojas de estilo CSS y un archivo de JavaScript.

Empezaremos por los enlaces que nos son necesarios para que nuestro proyecto funcione adecuadamente. Estos enlaces serán a las librerías de jQuery y las librerías de Bootstrap, por lo que todas las páginas lo tendrán. Sin estos enlaces no podríamos acceder a los recursos que se encuentran en dichas librerías y la página quedaría inservible ya que no se realzarían las acciones asignadas. También están los enlaces a los archivos CSS en los cuales hemos definido los colores y aspectos visuales de nuestra página. Así mismo, es necesaria una línea de código de llamada Bootstrap para que este se active y funcione correctamente. Los enlaces de archivos CSS y la llamada a Bootstrap se sitúan en la cabecera o *Header* de nuestra página, mientras que las referencias a las librerías de jQuery se situarán dentro del *body* de la propia página.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximun-scale=1.0 minimum-scale=1.0">
    <link rel="stylesheet"  href="css/bootstrap.min.css">
    <link rel="stylesheet"  href="css/estilos.css">
</head>
```

Figura 14. Enlaces CSS en la cabecera de HTML.

```
    <script src="http://code.jquery.com/jquery-latest.js"></script>
    <script src="js/bootstrap.min.js"></script>
</body>
</html>
```

Figura 15. Enlaces a jQuery, dentro del body de HTML.

Esto será así para todas las secciones de la página menos para la tercera, donde se desarrolla la videoconferencia. En este caso tendremos que añadir alguna llamada más los archivos JavaScript los cuales utiliza esta página para poder desarrollar el sistema de videoconferencias, así como a otra hoja de estilos CSS propia de esta aplicación. Las demás secciones no necesitan estos enlaces porque no tienen que recurrir a ningún archivo JavaScript para realiza su cometido. Estos enlaces los colocaremos en el *Header* también.

```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximun-scale=1.0 minimum-scale=1.0">
    <link rel="stylesheet"  href="css/bootstrap.min.css">
    <link rel="stylesheet"  href="css/estilos.css">
    <link rel="stylesheet" href="style.css">
    <script src="//simplewebrtc.com/latest.js"></script>
    <script src="ps-webrtc-simplewebrtc-final.js"></script>
</head>
```

Figura 16. Enlaces de CSS y JavaScript de la tercera sección.

Todas las secciones tendrán un menú de cabecera en el cual podremos ir a las restantes secciones así como unos links de interés en los que se enlaza nuestra página web con las webs oficiales de las tecnologías usadas para el desarrollo de este proyecto. Estas son: WebRTC, Bootstrap, JavaScript y jQuery. De este modo la navegación será fácil y podremos desplazarnos de una sección a otra o acceder a los links ya mencionados en cualquier momento.

Figura 17. Menú de navegación.

4.2 RESPONSIVIDAD

El objetivo del proyecto era que la página fuera responsiva y se visualizara de manera correcta en las diferentes resoluciones de pantalla de los distintos dispositivos de visualización, para que el mismo contenido se pueda observar bien tanto en un monitor de ordenador de mesa como en el dispositivo móvil más pequeño del mercado.

Para poner los ejemplos utilizaremos el menú de navegación, que está presente en todas las páginas y el contenido de la segunda sección donde hay bastante texto y cambia su formato de presentación según donde lo visualicemos.

Empezaremos por el menú.

Figura 18. El menú para pantallas grandes.

A medida que las resoluciones de pantalla se hacen más pequeñas el texto que hay en el menú se amontona y al no tener espacio no se visualizaría correctamente, por lo que para dichas resoluciones es necesario poder ocultar el menú añadiendo un botón para su posterior despliege en caso de que deseemos usarlo. A continuación se añadirán dos ejemplos de cómo responde el menú de navegacion ante diferentes resoluciones, Para eso vamos a emplear el ejemplo de visualizarlas en un iPad y posteriormente en un móvil. Si lo visualizaramos en la monitor de un ordenador obtendríamos los mismos resultados que en el iPad pero con algo más de tamaño.

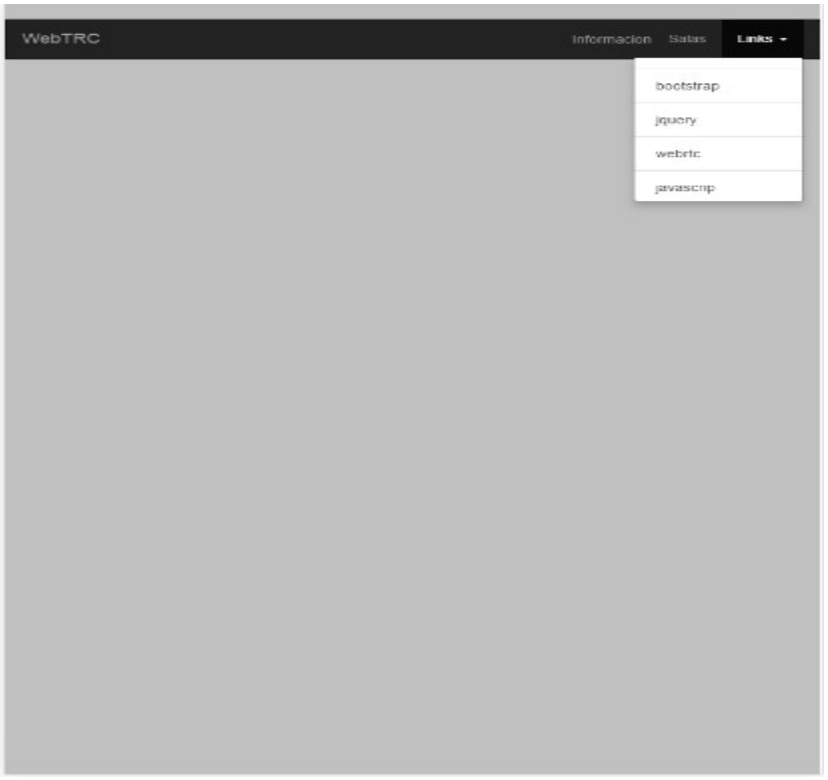
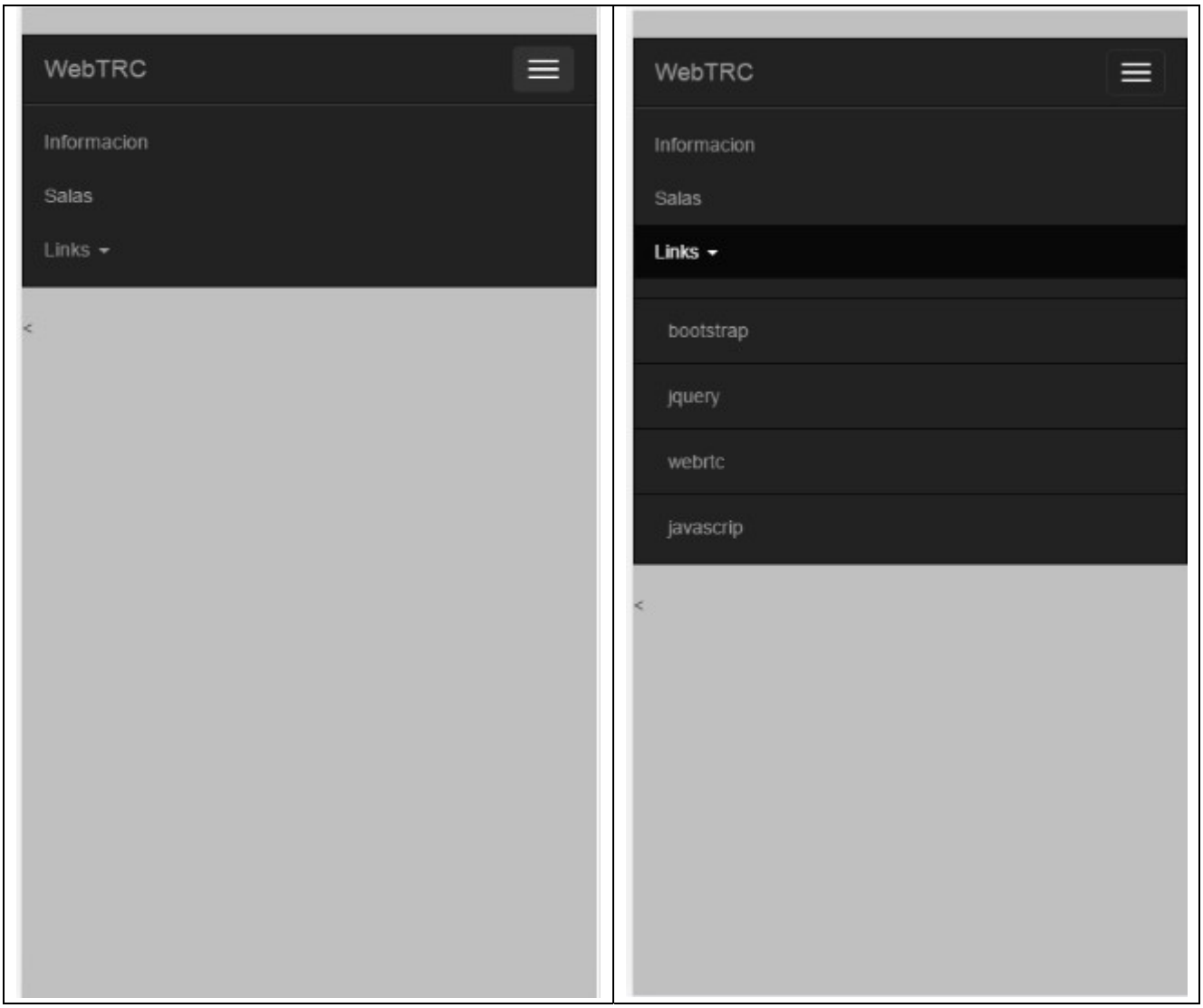


Figura 19. El menú en un iPad.

Figura 20.El menú en un móvil.

El contenido del menú queda oculto a simple vista, pero pulsando el botón se despliega para poder ser utilizado.



Figuras 21 y 22. Apertura del menú oculto.

Con el contenido de las secciones pasa lo mismo, según donde se visualice el contenido se mostrara de una forma diferente.

A continuación veremos la segunda sección de nuestra página y cómo responde ante diferentes tamaños de dispositivos

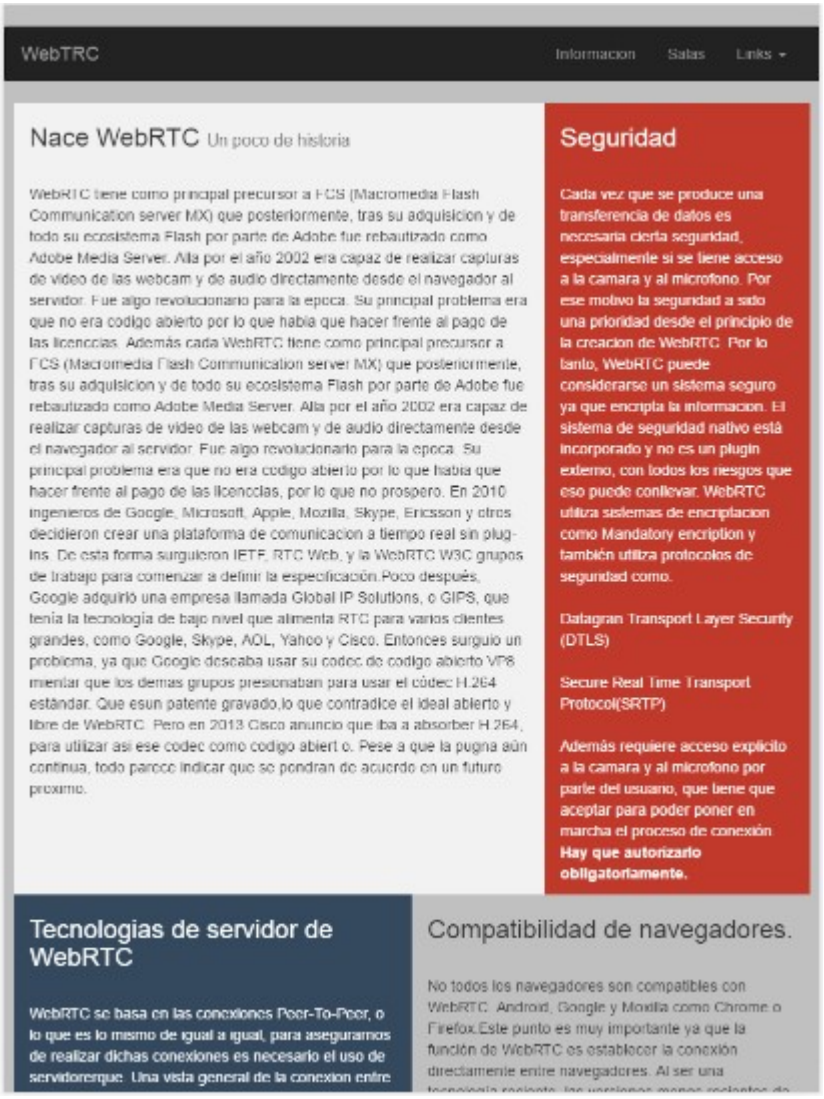


Figura 23. Contenido visualizado en un iPad.

Puesto que el contenido de texto de esta sección es superior a lo que se visualiza en pantalla, añadiremos otra figura en la cual veremos lo que se visualizaría si dentro del propio iPad se desplazara el contenido para leer el contenido inferior.

Figura 24. Se ve cómo quedaría en un iPad desplazando el contenido hacia abajo para visualizar el resto de la página.

A la hora de visualizarlo en un móvil se cambia el orden en el que aparece el contenido de dicha sección con el fin de que se pueda leer lo más claramente posible en pantallas tan pequeñas.

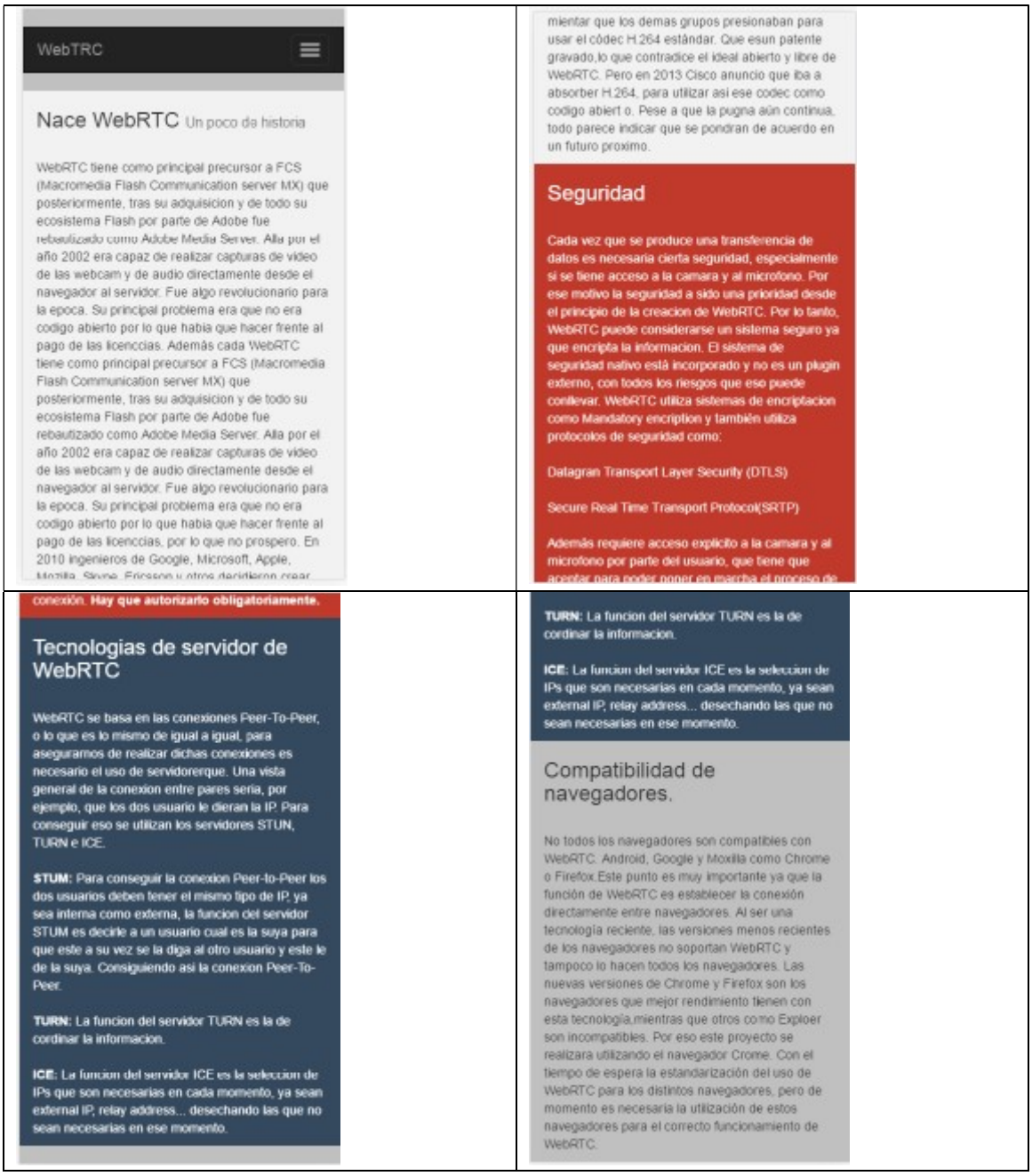


Figura 25. Se muestra cómo responde ante un móvil. Al igual que antes se visualiza como si moviéramos la página hacia abajo para leer todo su contenido.

Como se puede observar al ser visualizado con un móvil el contenido de nuestra sección, se ha situado uno encima del otro y no al lado como sí está en el tamaño iPad. Mediante la hoja de estilos CSS le hemos añadido color para que se observe más claramente los diferentes contenidos de esta sección cuando se sitúan un artículo encima del otro.

En la última sección, donde se crea la sala de videoconferencia, también actúa de forma responsiva.

Figura 26. La última sección en tamaño iPad.

Figura 27. Última sección en tamaño móvil.

4.3 FUNCIONAMIENTO SALA VIDEOCONFERENCIA

En la tercera y última sección, donde tiene lugar la sala de videoconferencias, el funcionamiento será el siguiente:

- Primero se nos pedirá que nombremos la sala que queremos crear, dándole el nombre que nosotros queramos.

Figura 28. 1. Paso para crear la sala.

- Si se le da a crear la sala sin haber introducido ningún nombre, la página creará la sala con un nombre aleatorio.

Figura 29.Obtencion de la dirección aleatoria de nuestra sala.

- Una vez nombrada la sala, se nos proporcionara una dirección http, para que se la entreguemos a quien queramos que se conecte a nuestra sala.

- Cuando el segundo usuario quiera acceder a la sala previamente creada, únicamente tendrá que poner la dirección http, proporcionada por el creador de la sala, en su navegador para establecer la conexión y realizar de ese modo la videoconferencia.

Nuestra tercera sección de la página cuenta con su archivo HTML, dentro de este archivo encontramos la etiqueta *<form>* que crea un formulario para que introduzcamos el nombre de la sala que hemos elegido, al introducir el nombre, se obtienen datos que son

utilizados posteriormente por el archivo de JavaScript, que nos devolverá la sala con nuestro video y el del otro usuario cuando se conecte.

```
    <h1 id="title">Bienvenido a tu propia sala de videoconferencia</h1>
   <br><br>
<form id="createRoom">
 <input id="sessionInput"/>
 <button type="submit">Ponle nombre a tu sala y creala</button>
   </form>
   <br><span id="subTitle">Si no le pones nombre a la sala se le asignara uno aleatoriamente</span>


<div id="roomLink" style="font-size:18px; color:#F00;"></div>
<video id="localVideo" width="250" style="margin-top:25px;"></video>
<div id="remotes"></div>
<br><br><br><br>
```

Figura 30. Parte del código HTML de la tercera sección.

El funcionamiento del archivo se JavaScript es el siguiente:

Primero establece la conexión WebRTC, después crea los elementos que contendrán el vídeo local y el vídeo remoto. Cuando se ha establecido la conexión WebRTC correctamente, nos muestra nuestro video local que es lo que graba la webcam. (Figura 28.) Utilizando el formulario de página HTML se crea la sala con el nombre que el nombre dado o uno aleatorio en el caso de que se ejecute el formulario sin haber introducido nombre alguno y nos devuelve la dirección de nuestra sala. Cuando otro usuario se conecta graba su propio archivo local (otra vez será lo que grabe su webcam) y como entra directamente a una sala ya creada por el usuario uno, los dos usuarios se conectan y empieza la transferencia de archivos, por lo que se nos muestran en pantalla los dos videos d los dos usuarios (Figura 27.) que son los archivos que transferimos.

```
function setRoom(name) {
    $('form').remove();
    $('h1').text('bienvenido a la sala: ' + name);
    $('#subTitle').text('dale esta direccion a tus amigos para que entren en tu sala:');
    $('#roomLink').text(location.href);
    $('body').addClass('active');
}

    if (room) {
    setRoom(room);
} else {  // If not, create one when the user submits the form
    $('form').submit(function () {
        var val = $('#sessionInput').val().toLowerCase().replace(/\s/g, '-').replace(/[^A-Za-z0-9_\-]/g, '');
        webrtc.createRoom(val, function(err, name) {
            var newUrl = location.pathname + '?' + name;
            if (!err) {
                history.replaceState({foo: 'bar'},null, newUrl);
                setRoom(name);
            }
        });
        return false;
    });
}
```

Figura 31. Parte del archivo de JavaScript. Fncion que recoge el nombre introducido y crea la sala

Cabe destacar que todas estas acciones se realizan mediante el archivo de JavaScript que está implementado por WebRTC. Al ser de código abierto dicho archivo está disponible al público en general y puede ser modificado, como se ha hecho en este caso.

5. CONCLUSIONES

La conclusión principal de este proyecto es que con WebRTC se abre un amplio abanico de posibilidades para los desarrolladores web. Ya que se podrán desarrollar multitud de aplicaciones de una manera sencilla gracias a las API de WebRTC, sin la necesidad de instalar *plugin* o de preocuparnos por distintas compatibilidades.

Al tratarse de código abierto cualquiera puede descargarse los archivos de JavaScipt y modificarlos a su gusto consiguiendo aplicaciones de compleja implementación con solo unas nociones de programación. Por lo que la principal conclusión de realizar este proyecto es que se crearán multitud de aplicaciones para la comunicación en tiempo real conforme se vaya conociendo WebRTC y extendiendo su uso.

Pese a que todavía se encuentra en fase de desarrollo el hecho de que sea software de código abierto hará que su desarrollo sea más acelerado, ya que las aplicaciones de código abierto suelen tener una gran comunidad detrás que comparte los avances conseguidos y proporciona ayuda a nuevos desarrolladores, mediante el uso de diferentes blogs, foros…

6. LÍNEAS FUTURAS

En este proyecto nos hemos centrado en conocer WebRTC, ver cómo trabaja viendo su funcionamiento interno de manera teórica y realizar un sencillo ejemplo práctico con la creación de la sala de videoconferencia.

Se podría decir que hemos visto el potencial de WebRTC, pero a medida que pase el tiempo se desarrollarán nuevas funciones. Por lo que las mejoras que se pudieran implementar en este proyecto son varias. En la misma sala de videoconferencia se podría añadir una ventana con chat de texto para que los usuarios de la sala puedan escribir. También se pueden poner opciones para poder minimizar u ocultar las ventanas de vídeo donde sale el propio usuario y dejar que se visualicen las del resto de usuarios para verlas mejor. Además existe la posibilidad de implementar una especie de pizarra donde poder dibujar y que el resto de usuarios pueda ver a la vez que tú lo estás dibujando, mientras lo estás dibujando claro. Compartir el escritorio es otra opción que puede realizarse ya que existen aplicaciones de escritorio remoto.

El recorrido de WebRTC es corto, apenas lleva dos años en funcionamiento por lo que es de esperar que conforme se vaya expandiendo su uso a cada vez más usuarios estos desarrollen nuevas capacidades y acciones con las que implementar nuestra sala de videoconferencias.

7. BIBLIOGRAFÍA

7. Bibliografía

[1]- Wikipedia ---------- https://es.**wikipedia**.org/

[2]- Youtube---------- https://www.**youtube**.com/

[3]- Página oficial de WebRTC---------- https://**webrtc**.org/

[4]- Página sobre tecnologías informáticas------ www.**w3schools**.com/

[5]- Página oficial de JavaScript---------- https://www.**javascript**.com/

[6]- Página oficial de jQuery---------- https://**jquery**.com/

[7]- Página oficial de Node.js---------- https://**nodejs**.org/

[8]- Página sobre HTML---------- **html5**facil.com/

[9]- Página oficial de CSS---------- www.**css3**.info/

[10]- Página oficial de Bootstrap---------- get**bootstrap**.com/

[11]- Curso sobre WebRTC de Lisa Larson Kelley